MODCOMP

MAX IV Programmer's Reference Manual

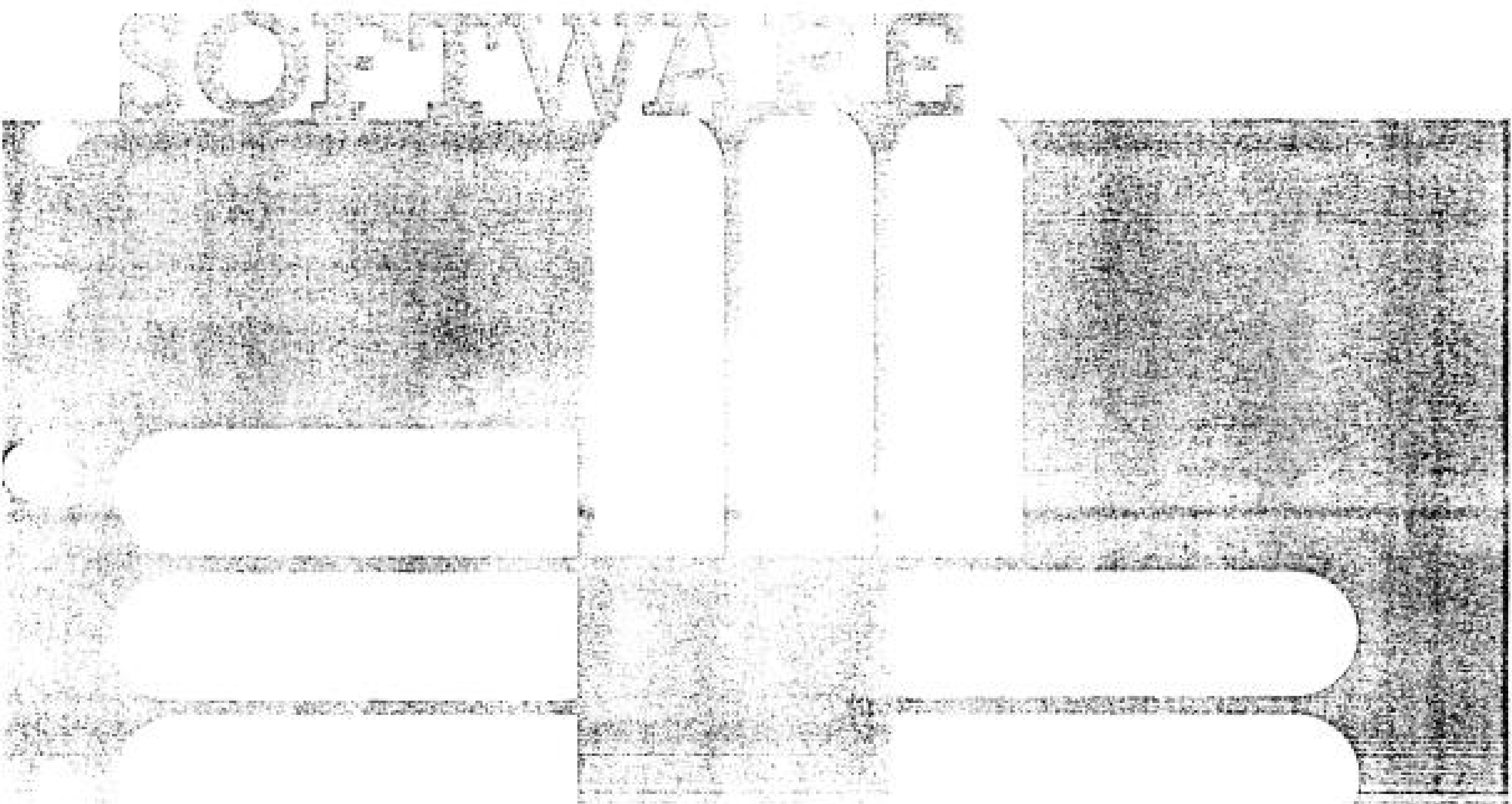
Link Editor (EDI)

211-804009-O00

MODCOMP
MODULAR COMPUTER SYSTEMS, INC

Programmer's Reference Manual MAX IV

Link Editor (EDI)



Manual History

Manual Order Number: 211-804009-O00
Title: *MAX IV Programmer's Reference Manual, Link Editor (EDI)*
Product Number: 600599-105O.0

Revision Level	Date Issued	Description
—	07/76	Initial Issue.
—	04/78	Revision. Changes for N.1.
—	07/81	Reissue. Changes for N.2.
—	03/82	Reissue. Changes for N.3.
—	10/82	Revision.
—	09/84	Reissue. This issue includes a users index. This revised manual reflects an update to include MAX IV only.
O00	04/85	Revision (O.0). Added Software Part Number for MAX IV Link Editor (EDI).

PREFACE

Audience: This manual is addressed to the system programmer and system operator using the Link Editor (EDI).

Subject: This manual contains detailed reference information and complete descriptions of the directives associated with the Link Editor (EDI).

Product Support: This system processor runs under the MAX IV Operating System.

Special Symbols and Notations: A revision bar (|) located in the margin of the page in the text indicates a change to the manual. This change generally represents a technical change to the product due to product revision. A revision bar is also entered in the Table of Contents to flag the general location of changes in the text.

Symbols used in defining syntax include:

xxx	Lower-case letters represent a variable, the value of which is supplied by the user.
XXX	Upper-case keywords are to be entered as shown.
[ ]	Brackets indicate either insignificant characters or an optional parameter.
{ }	Braces indicate a set of options from which the user is required to choose at least one.
...	An elipsis indicates a continuing sequence.

Related Publications: The reader is referred to the following manuals for additional information. The manual order numbering system has been revised.

Order Number	Manual Title
213-804001-REV	MAX IV GENERAL OPERATING SYSTEM System Guide Manual
212-804001-REV	MAX IV DATA STRUCTURES System Design Manual
211-804002-REV	MAX IV/MAX 32 NONRESIDENT JOB CONTROL AND BATCH FACILITIES Programmer's Reference Manual
211-804011-REV	MAX IV TASK/OVERLAY CATALOGER (TOC) Programmer's Reference Manual

210-804001-REV	MODCOMP ASSEMBLERS Language Reference Manual
211-804008-REV	MAX IV/MAX 32 Library Update Programmer's Reference Manual

When ordering manuals, use the Manual Order Number listed above. The latest revision (REV) will be shipped.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

CHAPTER 1
OVERVIEW OF LINK EDITOR (EDI)

1.1 PURPOSE OF THE LINK EDITOR (EDI)

The Link Editor (EDI) is a program that performs the following functions under the MAX IV Operating System.

1. Creates complete binary load modules from incomplete Assembly Language or FORTRAN programs.

2. Creates complex overlay load modules.

3. Recognizes global common areas.

NOTE: **The object code cannot contain any function codes unique to the MODCOMP CLASSIC Macro Assembler. These function codes are generally produced by the use of counters and attributes. (Refer to the MODCOMP ASSEMBLERS, Language Reference Manual.)**

It is essential that the user in a MAX IV environment utilize the MAX IV Link Editor (M4EDIT) whenever possible. One case that is not possible to use the MAX IV Link Editor (M4EDIT) is during the link-edit phase of system generation (SYSGEN). Only the Link Editor (EDI) produces output compatible with the Stand-Alone Linking Loader (SAL).

The Link Editor program name is EDIT and is often cataloged under EDI when a disc is available. This shorter name is referenced in the following sections.

1.2 LOGICAL FILES USED BY JOB CONTROL

Control Input (CI)	EDI directives are read from the CI file when the AO option is reset.
Alternate Control Input (AI)	EDI directives are read from the AI file when the AO option is set.
Input File for Main or Primary Object Programs (xx)	The xx file is specified by the user at editing time through the EDIT Directive. Main object program(s) are read from the specified file. When a file mark is encountered, a search of the user's private library (if any) and of the System Library (LB) is started.
User Library (UL)	In addition to the standard System Library file LB, the user can create a private library. The UL file is assigned to the device on which this private library exists. At editing time, EDI first searches the UL

file for missing External names and then switches this search to the LB file. If UL was assigned to NO or was not assigned at all, then EDI assumes that there is no user's private library and proceeds to search the LB file.

When UL is assigned to a magnetic tape or a disc partition, EDI reads the main object program from the xx file until a file mark is encountered. At this point, the UL file is searched in exactly the same manner as the LB file is searched (refer to System Library (LB) file search below). If more modules are needed after UL has been searched completely, the search is switched to the LB file.

System Library (LB)

The LB file is normally assigned to the device that holds the System Library.

This is the last file that is searched by EDI. The LB file is rewound before every file search for missing External names is initiated. Object modules are read from this file, and if all the modules with the missing External names are found, then pass one of the Link Editor comes to an end.

The LB file is rewound and searched again if at least one module was loaded during the previous search and not all missing External names were found. If no modules were loaded during a search, then a Link Editor (EDI) message "MISSING ROUTINES" is written on the CO file (refer to Appendix A).

Scratch File (SC)

When the SC option is set, the main and other modules loaded from UL and LB are written on the SC file. EDI reads all these modules from this SC file (after rewinding it) during pass two.

Listing Output (LO)

The entered directives and Map listing of the symbol table are written on the LO file.

Binary Object (BO)

The object code for the resulting Load Module is written on the BO file.

Communication (CO)

The Link Editor's error messages are written on the CO file. This is normally a global file.

Figure 1-1 (below) is a graphic representation of the files described in this section.

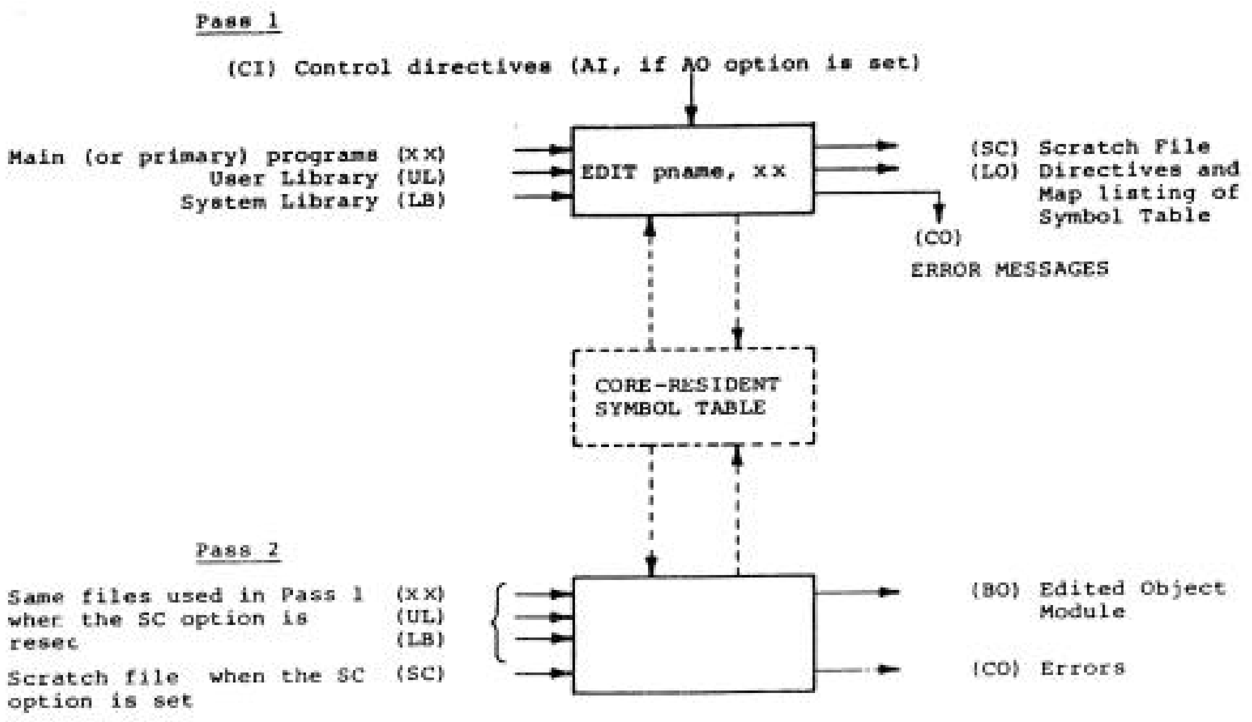

Pass 1
(CI) Control directives (AI, if AO option is set)
Main (or primary) programs (XX)
User Library (UL)
System Library (LB)
EDIT pname, XX
(SC) Scratch File
(LO) Directives and Map listing of Symbol Table
(CO)
ERROR MESSAGES
CORE-RESIDENT SYMBOL TABLE
Pass 2
Same files used in Pass 1 when the SC option is reset
(XX)
(UL)
(LB)
Scratch file when the SC option is set
(SC)
(BO) Edited Object Module
(CO) Errors

Figure 1-1. EDIT Directive Function

CHAPTER 2
SUMMARY OF DIRECTIVES

The following list is a summary of the Link Editor (EDI) directives detailed further in Chapter 3 of this manual.

ABS	Specifies addresses to become absolute
ASSIGN	Specifies logical files or devices
AVFILE	Advance file specified file-marks
AVRECORD	Advance file specified records
BKFILE	Move backward specified file-marks
BKRECORD	Move backward specified records
CLOSE	Temporarily suspends internal and COMMON block definitions
DEFX	Allows external definition usage at link edit
EDIT	Invokes processor to search for file
EXIT	Load and return to Job Control
GCOM	Establish global COMMON areas
INITIALIZE	Remove entries, associations and symbols
LEVEL	Specifies overlay level
LOCALL	Activates load-on-call
MAP	Write a MAP on LO file
MULTIPLE	Reads one or more load modules
$NOP	Display user comment
NOXEQUATE	Removes external associations
OPEN	Allows access to definitions
ORDER	Orders or allocates COMMON blocks
PAUSE	Display message and wait
PRIMARY	Reads one object module
REMOVE	Delete table entry
RESTORE	Copy symbol file

REWIND	Position device to beginning
SAVE	Copy current symbol file
TASK	Specifies resident foreground task
WEOF	Write end-of-file mark
XEQUATE	Specifies extended associations

CHAPTER 3
DIRECTIVES

The Link Editor (EDI) reads and performs directives. The following directives perform different functions that include:

- o utility functions
- o setting functions for the operation mode of subsequent directives
- o editing functions

The $ASSIGN, $REWIND, $AVFILE, $BKFILE, $AVRECORD, $BKRECORD, and $WEOF are equivalent Job Control Directives. Refer to the MAX IV/MAX 32 NONRESIDENT JOB CONTROL AND BATCH FACILITIES, Programmer's Reference Manual for more information. The special symbols are explained in the Preface.

ABS

SPECIFIES ADDRESS TO BECOME ABSOLUTE

The ABS Directive causes the addresses of all Internal and External names in the current symbol table to become absolute. Use the ABS Directive followed by a SAVE Directive to save the system's symbol table after link-editing.

SYNTAX

```
     ABS
```

SPECIFIES LOGICAL FILES OR DEVICES

The ASSIGN Directive assigns logical files to logical devices or to other logical files. A logical file assigned to itself is assigned to its default assignment in the system.

SYNTAX

```
                           1      2
     ASS[IGN]            file name...
```

file	Parameter 1 (required) specifies a file name to be assigned.
name...	Parameter 2 (required) represents the name of a device or file.

EXAMPLES

```
     ASS BI TY
     ASSIGN,LO=BI,BO=DNA
```

ADVANCE FILE SPECIFIED FILE-MARKS

The AVFILE Directive advances a device medium in the forward direction until the specified number of file marks have been read.

SYNTAX

```
                          1   2
     AV[FILE]           file [n]
```

file	Parameter 1 (required) represents the file to be advanced.
[n]	Parameter 2 (optional) specifies the number of file marks to be advanced. (Default) If Parameter 2 is not specified, a value of 1 is assumed.

EXAMPLES

```
     AVF SI
     AVFILE,SI=7
     AVF BO,9
```

ADVANCE FILE SPECIFIED RECORDS

The AVRECORD Directive moves a device medium in the forward direction until the specified number of physical records have been skipped.

SYNTAX

```
                        1   2
     AVR[ECORD]        file [n]
```

file	Parameter 1 (required) represents the file to be advanced.
[n]	Parameter 2 (optional) represents the number of records to be advanced (skipped). (Default) If Parameter 2 is not specified, a value of 1 is assumed.

EXAMPLES

```
     AVR SI
     AVRE BO=9
```

BKFILE

MOVE BACKWARD SPECIFIED FILE-MARKS

The BKFILE Directive moves a device medium in the reverse direction until the specified number of file marks have been reached.

SYNTAX

```
                              1   2
     BKF[ILE]                file [n]
```

file	Parameter 1 (required) represents a file that is to be moved backwards.
[n]	Parameter 2 (optional) represents the number of file marks to be backspaced. (Default) If Parameter 2 is not specified, a value of 1 is assumed.

EXAMPLES

```
     BKF SO
     BKF,BO,8
```

MOVE BACKWARD SPECIFIED RECORDS

The BKRECORD Directive moves a device medium in the reverse direction until the specified number of physical records have been skipped.

SYNTAX

```
                         1   2
     BKR[ECORD]          file [n]
```

file	Parameter 1 (required) represents a file name.
[n]	Parameter 2 (optional) represents the number of records to be backspaced. (Default) If Parameter 2 is not specified, a value of 1 is assumed.

EXAMPLES

```
     BKR SO
     BKR,SI,3
```

TEMPORARILY SUSPENDS INTERNAL AND COMMON BLOCK DEFINITIONS

The CLOSE Directive is useful in building overlays. It temporarily suspends use of all Internal and Common Block definitions within the specified programs for all of the following EDIT functions until an OPEN Directive is encountered.

SYNTAX

```
                       1
     CLO[SE]          name-1...
```

name-1...	Parameter 1 (at least one is required) represents main program names.

EXAMPLES

```
     CLOSE ABLE
     CLOSE MAIN,DOG
```

DEFINE EXTERNAL REFERENCE

The DEFX Directive allows the definition of External references at link-editing time by the user. If the quantity "address" contains a dollar mark in its prefix, its value is assigned to "name" as a relocatable quantity. If the dollar mark is not present, "address" is used as an absolute quantity.

NOTE: The character # preceeding numeric characters within the directive signifies a hexadecimal value.

SYNTAX

```
                     1    2
     DEF[X]         name address
```

name	Parameter 1 (required) represents a symbolic name (up to six characters long) that appears in some object modules as an External reference.
address	Parameter 2 (required) represents a memory address and can be a relocatable or an absolute quantity.

EXAMPLES

```
     DEF M$A,2304
     DEFX M4B,#3245
```

START LINK EDITING

The EDIT Directive causes the processor to search the specified file, starting from its current position, for the specified program name, Parameter 1. When it is found, the program is loaded and link-edited. If MAIN is entered, the first object module that is read from Parameter 2 is link-edited, regardless of the program name specified within that module.

The MULTIPLE and PRIMARY Directives are used to set up the mode of operation for the EDIT Directive, when editing overlays. Figure 1-1 shown in Section 1.2 presents a general picture of the EDIT Directive function. If the editing level is currently zero (see the LEVEL Directive), the EDIT Directive causes the current symbol table to be set to the initialized state.

SYNTAX

```
                          1       2
     EDI[T]              [MAIN  ] file
                         [pname]
```

[MAIN] [pname]	Parameter 1 (at least one is required) represents a program name. (Default = MAIN)
file	Parameter 2 (required) represents a file name.

EXAMPLES

```
     EDI MAIN,SO
     EDIT FOX,BI
     EDI,DOG,SI
```

EXIT LINK EDITOR

The EXIT Directive causes the Link Editor to exit and the operating system to load and transfer control to Job Control.

SYNTAX

```
     EXI[T]
```

EXAMPLE

```
     EXIT
```

ESTABLISH GLOBAL COMMON AREAS

Global common areas are defined and allocated real memory at system generation (SYSGEN). A global common area can be inserted into a task's virtual space at load time (by using the Task/Overlay Cataloger INSERT Directive) or dynamically during execution by calling the proper REX Service. Refer to the MAX IV GENERAL OPERATING SYSTEM, System Guide Manual.

In order to link a local common block to a global area the user must know at link-edit time where the global area is inserted in the task's virtual space. By using this address in the GCOM Directive all references to the local block become references to the global area when the task is executing (and if all the global area has been inserted).

SYNTAX

```
                          1        2      3
     GCO[M]             cbname address size
```

cbname	Parameter 1 (required) represents a symbolic name given to a common block label.
address	Parameter 2 (required) represents a memory address (see DEFX Directive).
size	Parameter 3 (required) represents the maximum size (in computer words) of the global area.

EXAMPLES

```
     GCOM BLOCK,#100,#1000
     GCO BLOCK,#100,4096
```

REMOVE ENTRIES, ASSOCIATIONS AND SYMBOLS

The INITIALIZE Directive removes all load-on-call entries, associations made by the XEQUATE directives, and symbols currently in the symbol table and returns the Link Editor (EDI) to the initialized state. The Link Editor (EDI) is automatically initialized when it is loaded by the system.

SYNTAX

```
     INI[TIALIZE]
```

EXAMPLES

```
     INIT
     INITIALIZE
     INI
```

LEVEL

SPECIFIES OVERLAY LEVEL

The LEVEL Directive is used in building overlay modules. The user can use up to 1023 overlay levels and can return to any previously established level by entering a LEVEL Directive containing the same level number.

The zero (0) level is a special case that resets the relocation bias to zero (0) for all following EDIT Directives.

If Parameter 2 is specified, then the start address for this level coincides with the address of the Internal name in Parameter 2. If Parameter 3 is specified, the start address for the level is this relative address. If Parameter 4 is specified, the start address for the level is the location after the highest address used in editing the previous module.

SYNTAX

```
                          1       2        3        4
     LEV[EL]          number [name] [address] [*]
```

number	Parameter 1 (required) represents the overlay level and can be any decimal value between 1 and 1023.
[name]	Parameter 2 (optional) represents the name of a previously defined Internal Definition.
[address]	Parameter 3 (optional) represents a hexadecimal address.
[*]	Parameter 4 (optional) represents the next location after the previous module edited.

EXAMPLES

```
     LEVEL 1
     LEVEL 5
     LEVEL 1,*
     LEVEL 3,OVER3
     LEVEL 2,OVER2
     LEVEL 1,#3234
     LEVEL 5,#5123
```

WRITE A MAP ON THE LO FILE

The MAP Directive causes a Map of the current symbol table to be written on the LO file. If the level is zero (Ø), then the symbol table is cleared on the next EDIT Directive.

SYNTAX

```
MAP
```

EXAMPLE

```
MAP
```

MULTIPLE

READS ONE OR MORE OBJECT MODULES FROM xx

The MULTIPLE Directive causes the Link Editor (EDI) to read one or more object modules from the xx file. The user can follow the main program to be edited by additional modules that are to be read from the same xx file in trying to satisfy missing Externals. The search for any missing External(s) switches to one of two files (UL and/or LB) when a file mark is read from xx.

MULTIPLE is the default mode when the Link Editor (EDI) is executed.

SYNTAX

```
     MUL[IPLE]
```

EXAMPLE

```
     MULTIPLE
```

DISPLAY USER COMMENT

The $NOP Directive displays user comment.

SYNTAX

```
     $NOP
```

EXAMPLE

```
     $NOP  This section inputs data from DEPT 509.
```

NOXEQUATE

REMOVES EXTERNAL ASSOCIATIONS

The NOXEQUATE Directive removes all associations of External names previously entered by means of an XEQUATE Directive.

SYNTAX

```
NOX[EQUATE]
```

EXAMPLES

```
NOX
NOXEQUATE
```

ALLOWS ACCESS TO DEFINITIONS

The OPEN Directive causes all Internal and Common Block definitions in the specified programs (name-1) to be accessible to subsequent EDIT functions. This feature of the Link Editor (EDI) prevents the unnecessary loading of function subprograms that are already in memory by an overlay that is executed on a level different from the one these functions were loaded under.

SYNTAX

```
                         1
    OPE[N]           name-1...
```

name-1...	Parameter 1 (at least one is required) represents main program names.

EXAMPLES

```
    OPEN ABLE
    OPEN DOG,FOX
```

SPECIFY ORDERING OF COMMON BLOCKS

The ORDER Directive can be used to order the user's labeled common blocks in memory and/or to allocate a larger common block than that specified in the program.

SYNTAX

```
                          1       2           3          4
     ORD[ER]          name-1 [size-1] [,name-2 [size-2]]...
```

name-1	Parameter 1 (required) represents labelled common block names.
[size-1]	Parameter 2 (optional) represents common block sizes in words.
[name-2] [size-2]	Parameters 3 and 4 (optional) represent additional labelled common block names and common block sizes.

EXAMPLES

```
     ORDER A,100,B,500,C,D
     ORD ZAP,#1000,ZIP
```

DISPLAY MESSAGE AND WAIT

PAUSE can be used when the Link Editor (EDI) directives are entered through a device other than a terminal. It is intended to direct the operator to intervene at intermediate points during the link-editing. This directive causes the Link Editor (EDI) to write the entire directive (PAUSE message) to the CO file, and then to HOLD. The Link Editor (EDI) must be resumed manually to exit from the HOLD state.

SYNTAX

```
                        1
     PAU[SE]          message
```

message	Parameter 1 (required) represents a message directed to the operator's attention.

EXAMPLE

```
     PAUSE ATTENTION: MOUNT MAGTAPE ON MT2
```

READ ONE OBJECT MODULE

The PRIMARY Directive causes the Link Editor (EDI) to read only one object module from the xx file in subsequent editing operations.

SYNTAX

```
PRI[MARY]
```

EXAMPLE

```
PRIMARY
```

DELETE TABLE ENTRY

The REMOVE Directive removes the specified main program name (pname) and all associated subprograms from the symbol table, and then compresses the symbol table.

SYNTAX

```
                       1
     REM[OVE]          pname
```

pname	Parameter 1 (required) is the name of a previously edited MAIN program.

EXAMPLE

```
     REMOVE FOX
```

RESTORE SYMBOL FILE

The RESTORE Directive restores the symbol table from a specified file. This symbol table should have been previously saved by means of the SAVE Directive.

SYNTAX

```
                      1
     RES[TORE]        file
```

file	Parameter 1 (required) is the name of a file that is used to read a previously saved symbol table, terminated by an end-of-file mark.

EXAMPLE

```
     RESTORE SO
```

POSITION DEVICE TO BEGINNING

The REWIND Directive positions a device to the beginning of the media.

SYNTAX

```
                     1
     REW[IND]        file
```

file	Parameter 1 (required) is the name of the file to be rewound.

EXAMPLES

```
     REW BI BO
     REW,SO,BO
```

SAVE

COPY CURRENT SYMBOL FILE

The SAVE Directive (with no SOURCE option) writes the contents of the current symbol table onto the specified file. This copy of the symbol table could be restored at a later date in order to edit programs using this previously created symbol table. The user must write an end-of-file mark after the saved symbol table.

The SAVE Directive, with the SOURCE option entered, generates from the current symbol table a macro file and saves it on the specified file. The name of the macro file is then the name of the main object program. All Internal names in this file are assigned addresses.

The user can use an INSERT Macro Assembler Directive naming this file and referencing these names directly in a program.

SYNTAX

```
                          1         2
      SAV[E]              file [S[OURCE]]
```

file	Parameter 1 (required) is the name of a file that is used for saving (writing) the symbol table or macro file.
SOURCE	Parameter 2 (optional) is the current symbol table.

EXAMPLES

```
      SAVE SO
      SAVE BO,S
```

WRITE END-OF-FILE MARK

The WEOF Directive writes a file mark on each logical file name specified.

SYNTAX

```
                         1
     WEO[F]           file...
```

file...	Parameter 1 (at least one is required) is the file that the end-of-file mark is to be written to.

EXAMPLES

```
     WEO BO
     WEOF BO,SO,SC
```

SPECIFIES EXTERNAL ASSOCIATIONS

The XEQUATE Directive associates "name-1" with "name-2" (See Parameter 1) in the symbol table so that External references made to "name-1" are satisfied by Internal definitions of "name-2". If an Internal definition of "name-1" is encountered during the editing of an object module it is ignored. If "name-1" and "name-2" both appear as External references both are satisfied by Internal definitions of "name-2".

SYNTAX

```
                                1
     XEQ[UATE]       name-1 name-2...
```

name-1 name-2...	Parameter 1 (at least one set is required) is a pair of symbolic names.

EXAMPLES

```
     XEQ ALPHA=BETA
     XEQUATE ALPHA=BETA
```

CHAPTER 4
GUIDE TO THE USE OF MAX IV LINK EDITOR (EDI)

4.1 CREATE RELOCATABLE LOAD MODULES FROM INCOMPLETE PROGRAMS

An Assembly Language program that references names (or labels) that are all defined within itself (no External references) is a complete program. When this program is assembled, its binary object output (generated by an assembler) is referred to as the program's relocatable load module.

An Assembly Language program that references names not defined within itself (External labels) is an incomplete program. When this program is assembled, its binary output (generated by the assembler) does not constitute a complete relocatable load module because the memory addresses for the External names are not satisfied. The memory addresses are dummy addresses to be filled at a later date.

To determine the values of the missing memory addresses, this incomplete module is link-edited with other object modules. These other object modules contain Internal name definitions identical to the names referenced as Externals in the module. They can also be identical to the names referenced as Externals in any of the previous modules if subsequent modules have their own External label references.

When all of the names referenced as Externals are found, the following occurs:

- o The addresses of all the External names are placed into a symbol table.
- o A Map of all External symbols is written on the LO file at the end of the first pass.
- o A second pass of the Link Editor (EDI) produces a relocatable Load Module of the complete program.

The first object program encountered by the Link Editor (EDI) is normally referred to as the main program or primary program module. The main program is read from a user-specified file. To satisfy missing External names, object programs are read from one or two library files. These are the User subroutine Library (UL) and/or the system subroutine LiBrary (LB).

The link-editing process requires two passes over the input modules as follows:

Pass One - The primary object module (or main program) is read and a memory-resident symbol table of all Internal definitions, External references, and Common allocation is constructed.

All External references are then resolved from one of the following:

- o The specified input file
- o The optional User Library file (UL)
- o The System Library file (LB)

If a rewindable scratch file (SC) is available, all required modules are copied onto this file for use during pass two. All External references must be resolved during pass one. At the end of this pass the Link Editor (EDI) produces a Map of the symbol table on the LO file (unless the MAP option is reset). The validity of all data is checked during this pass.

The primary object module must be HELD for input files. When fully rewound it is positioned at the start of the input file and at the start of the primary object module.

Pass Two - During the second pass over the object modules the following occurs:

- o All references to External names are satisfied from the symbol table.
- o A self-contained relocatable load module is written on the Binary Output file BO.
- o The sizes of object records are reblocked if necessary to a size suitable for the output device (up to 256 bytes per record).

4.2 CONSTRUCTION OF COMPLEX OVERLAY LOAD MODULES

This feature permits large programs to be segmented into smaller and serially executable programs called overlays. Flexibility in generating overlay load modules is made possible by the use of the LEVEL, CLOSE, and OPEN Directives.

4.3 RECOGNITION OF GLOBAL COMMON AREAS

When used in conjunction with Task-Overlay Cataloger (TOC), the Link Editor (EDI) is used to equate a local common block to a global area.

CHAPTER 5
SYSTEM PROCESSOR FEATURES AND LIMITATIONS

5.1 PROCEDURE REFERENCING THE LINK EDITOR (EDI)

The following Job Control procedure can be invoked in the MAX IV Utility Procedure by using the $DO Directive defined below.

```
$PROCEDURE EDIT
$ASS BI=%2 BO=%3
$REW BI
$IFM %4 1
$OPTION NOMAP
$IFM %7 1
$OPTION NOSC
$IFM %8 1
$OPTION NOLO
$IFP %6 6
$EXE EDIT
$IFP %5 P=PRIMARY
EXIT %1,BI
WEOF BO
EXIT
$AVR CI=2
$ASS AI=%6
$EXE EDIT,,AO
$NOTE END OF EDIT
```

This procedure is used to link-edit an object module that references External names. The UL file is searched for missing External names only if it was assigned prior to the "$DO EDIT,..." Command. The LB file is always searched last for any missing External names.

SYNTAX

```
                1    2  3  4    5        6        7     8      9
     $DO       EDIT,pname,bi,bo[,nomap][,primary][,ai][,nosc][,nolo]
```

EDIT	Parameter 1 represents the procedure name.
pname	Parameter 2 represents the program name to be edited.
bi	Parameter 3 represents the device name to be edited.
bo	Parameter 4 represents the device name or file name where the object binary output is to be written. (The BO file is not rewound by this procedure and a file mark is written on it at the end of the object code.)
nomap	Parameter 5 signifies that if any character(s) are entered for this parameter then no map listing is written on the LO file.

primary	Parameter 6 specifies that if any character(s) are entered for this parameter then the PRIMARY directive is inserted as the first directive of the SYSGEN Editor.
ai	Parameter 7 represents the device or file from which the Link Editor (EDI) directives are to be read (if different from the CI file).
nosc	Parameter 8 specifies that if any character(s) are entered for this parameter, then the SC option is reset.
nolo	Parameter 9 signifies that if any character(s) are entered for this parameter, the LO option is reset.

5.2 DATA STRUCTURE

The Link Editor (EDI) reads binary object records in MODCOMP's standard object-language format except for those function codes unique to M5A. Refer to the MODCOMP ASSEMBLERS, Language Reference Manual. The input record size in bytes per record is normally 80, 200, or 256 depending on the input device. Binary output records are created by the Link Editor (EDI). Their record size is equal to the record size specified for the device on which they are to be written, but never exceeding 256 bytes per record.

5.3 LIMITATIONS

The MAX IV Link Editor (EDI) processes binary object records in MODCOMP's Standard Binary format (except for those function codes unique to M5A). (Refer to the binary output format produced by the MACRO Assemblers).

5.4 OPTIONS

HOLD - If the HOLD option is set, then EDI HOLDs before reading any directive. The user must manually resume for each directive EDI is to read.

LO - If the LO option is reset, the directives are not listed on the LO file.

AO - If the AO option is set, then the EDI directives are read from the AI file (instead of the CI file).

SC - If the SC option is reset, the object modules loaded during pass one are not written on the SC file. In such a case, during pass two, all these modules are read from the same devices that they were read from during pass one, and in the same order.

MAP - If the MAP option is reset, then the Map listing of the symbol table is bypassed.

Ø - If user option zero is set, the listed output device to which the LO file is assigned is considered to be 72 columns wide regardless of what it has been SYSGENed as.

APPENDIX A
LINK EDITOR (EDI) ERROR MESSAGES

Error messages are written on the CO file. If the Control Input device is not a terminal device, the Link Editor (EDI) enters the HOLD state after the writing of an error message. In such an event, the Link Editor must be resumed manually to allow a new directive to be processed. If the Control Input is a terminal device, there is no HOLD, so that a new directive can be entered immediately after the writing of an error message.

The following are data integrity errors that could be encountered during the reading of binary records:

MESSAGE	MEANING	RECOMMENDED ACTION
LINK EDIT ERROR..SQ	Sequence error.	Check sequence.
LINK EDIT ERROR..CK	Checksum error.	
LINK EDIT ERROR..FC	Illegal function code. results from improperly positioning or assigning the input files or from using countered code.	Enter an EXIT Directive to go back to Job Control. Check and obtain valid object records. Retry link-editing.
STATEMENT ERROR	The directive entered is not acceptable.	Re-enter a corrected directive (if a terminal device is used) or change the incorrect directive in the job stream.
DUPLICATE INTERNAL NAME	Two or more definitions of a name have been encountered by the editor.	Resolve the conflict and re-edit.
LINK EDIT ERROR..CM	A common name is not found in the symbol table or the common name size is in error.	Check and correct the common name and size. retry link-edit.
LINK EDIT ERROR..XT	An external name definition is not found in the symbol table.	Check and correct name. retry link-edit.
LINK EDIT ERROR..EF	A file mark has been read from the BI file and the program name specified by the edit directive was not found.	Enter an EXIT Directive to return to Job Control and, after correcting the error, retry the link-edit.

MISSING ROUTINES	One or more external definitions have not been satisfied.	Enter the Operator Communication Directive /RES MAP to obtain a map listing to determine which external definitions are unsatisfied. If a particular use of the edited program does not require the satisfaction of the external definitions, enter /RES go to procedure an edited object module with the unsatisfied external definitions linked to absolute location zero. If /RES is entered, then /RES GO is assumed.

INDEX

Fold

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 3624 FT. LAUDERDALE, FL 33309

POSTAGE WILL BE PAID BY ADDRESSEE

MODULAR COMPUTER SYSTEMS
1650 W. McNAB ROAD
P.O. BOX 6099
FT. LAUDERDALE, FLORIDA 33310

Attention: TECHNICAL PUBLICATIONS, M.S. #85

Fold

MODCOMP®

Please comment on the publication's completeness, accuracy, and readability. We also appreciate any general suggestions you may have to improve this publication.

If you found errors in this publication, please specify the page number or include a copy of the page with your remarks.

Your comments will be promptly investigated and appropriate action will be taken. If you require a written answer, please check the box and include your address below. ☐

Comments: ______________________________

Manual Title ______________________________

Manual Order Number ______________ Issue Date ______________

Name ______________ Position ______________

Company ______________________________

Address ______________________________

______________ Telephone (   ) ______________

MCS 148D (1/83)